

# C10A - Lenguajes y Paradigmas de Programación

## Lenguaje de programación

“

Los **lenguajes tipados fuerte** y **débil** se distinguen según si permiten o no violaciones de los tipos de datos una vez declarados.



”

## Lenguajes de tipado débil

En estos lenguajes no indicamos, la mayoría de las veces, el tipo de variable. Aquí podemos asignar, por ejemplo, un valor entero a una variable que anteriormente tenía una cadena. Pero, no solo eso, también podemos operar con variables de distintos tipos.

Su principal **ventaja** es que es mucho más rápido de desarrollar, pero una clara **desventaja** es que podemos cometer muchos más errores si no tenemos cuidado.



## Lenguajes de tipado fuerte

En estos lenguajes se nos obliga a indicar el tipo de dato al declarar la variable. Además, dicho tipo no puede ser cambiado una vez definida la variable.

La **ventaja** es que al ser código más expresivo, cometeremos menos errores.

La **desventaja** es que son mucho más estrictos a la hora de programar y que hay que escribir mucho más código.



## Lenguajes de tipado estático

En el tipado estático, la **comprobación** de tipificación se realiza **durante la compilación** y no durante la ejecución. Comparado con el tipado dinámico, el estático permite que los errores de tipificación sean detectados antes y que la **ejecución** del programa sea **más eficiente y segura**.

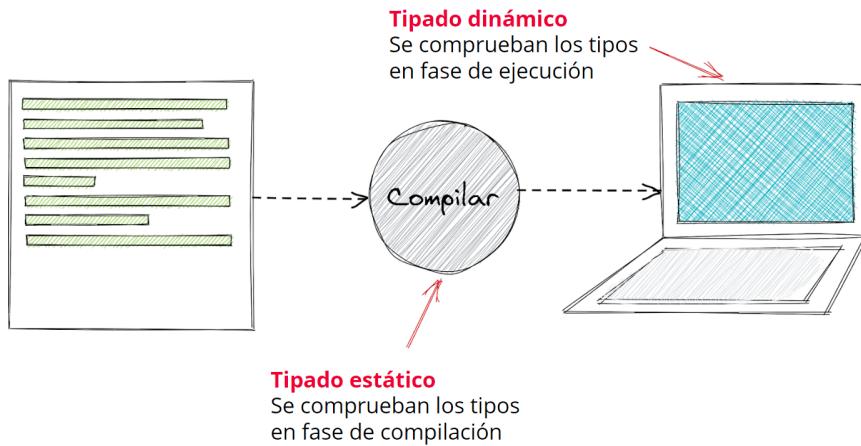


## Lenguajes de tipado dinámico

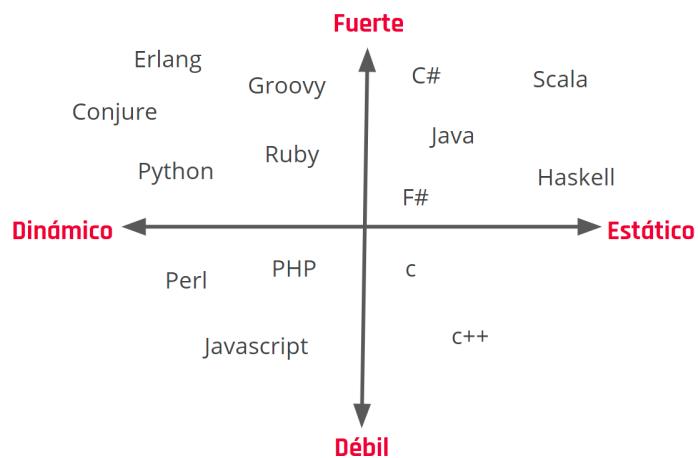
La comprobación de tipificación se realiza durante su ejecución en vez de durante la compilación. Comparado con el tipado estático, este es más flexible, a pesar de ejecutarse más lentamente y ser más propenso a contener errores de programación.



## Tipado dinámico y tipado estático



## Posicionamiento de cada lenguaje de programación



## Frameworks ~ Marco de Trabajo

Es una estructura previa / esqueleto que se puede aprovechar para desarrollar un proyecto.

El Framework es una especie de plantilla, un esquema conceptual, que simplifica la elaboración de una tarea, ya que solo es necesario complementarlo de acuerdo a lo que se quiere realizar.



Tipado de lenguajes y framework

DigitalHouse >  
Coding School

## Paradigmas de Programación

### ¿Qué es un paradigma?

Un paradigma es una forma de pensar bajo un modelo preestablecido.



### Paradigmas de programación

# Paradigmas de programación



- Paradigma estructurado**
- Paradigma de programación orientado a objetos**
- Paradigma funcional**
- Paradigma lógico**
- Paradigma de programación con lenguaje específico de dominio**
- Multiparadigma**

## **Paradigma estructurado**



Sigue una línea de pensamiento donde se suele ejecutar una instrucción a la vez y uno se rige en un acotado set de instrucciones.

Este paradigma es muy utilizado para el desarrollo de sistemas.



### **Ejemplo:**

Una función “esPar” recibe un número y devuelve el mensaje “verdadero”, si el número es par, y “falso”, si es impar.



## ✓ Paradigma de programación orientado a objetos

El código puede agruparse de tal forma que llegue a representar una entidad y que interprete mensajes. La fortaleza del paradigma de la programación orientada a objetos yace en utilizar abstracciones y crear entidades.

### Ejemplo:

Un código representa un carrito de compra.



Otro código representa un producto con su precio.



Luego, puedo agregarle la responsabilidad al carrito que vaya agregando productos para luego preguntarle el costo total.



## ✓ Paradigma funcional

El paradigma de programación funcional se basa en un concepto muy simple y es el de las funciones matemáticas.

La fortaleza de este paradigma radica en que siempre que a la función X se le pasa el valor A, esta siempre va a devolver el valor B. La fortaleza de este paradigma radica en que siempre que a la función X se le pasa el valor A, esta siempre va a devolver el valor B.



Esta propiedad de devolver el mismo valor se le conoce como inmutabilidad, y es característico de este paradigma.

### Ejemplo:

La solución funcional al problema de si un número es par o no es muy similar al estructurado, debemos crear una función "esPar" que reciba un número y nos diga si es par o impar.



## ✓ Paradigma lógico

En lugar de desarrollar pasos e instrucciones, utiliza reglas lógicas para consultar al sistema y el mismo infiere que hacer en base a las reglas lógicas establecidas.

Paradigma de programación lógica → Instrucciones → Reglas lógicas

### Ejemplo:

Reglas lógicas:

Toda persona cuyo saldo sea negativo es deudor.

A todo deudor se le aplica una tasa de interés del 10%

Con este set lógico podríamos preguntar:

¿Cuál es la tasa de interés de Juan?

El sistema responde analizando si Juan es una persona, si es deudor o no y si aplica o no la tasa de interés.



## ✓ Paradigma de programación con lenguaje específico de dominio

Los lenguajes que encontramos acá tratan de resolver problemáticas superespecíficas.

### Ejemplo:

Cuando queremos consultar una base de datos de un supermercado para saber qué productos tenemos en la categoría de electrodomésticos.



## ✓ Multiparadigma

A lo largo de la evolución de la programación, con nuevos desafíos y paradigmas han habido lenguajes que han modificado su estructura para poder permitir dar soluciones en distintos paradigmas.

### Ejemplo:



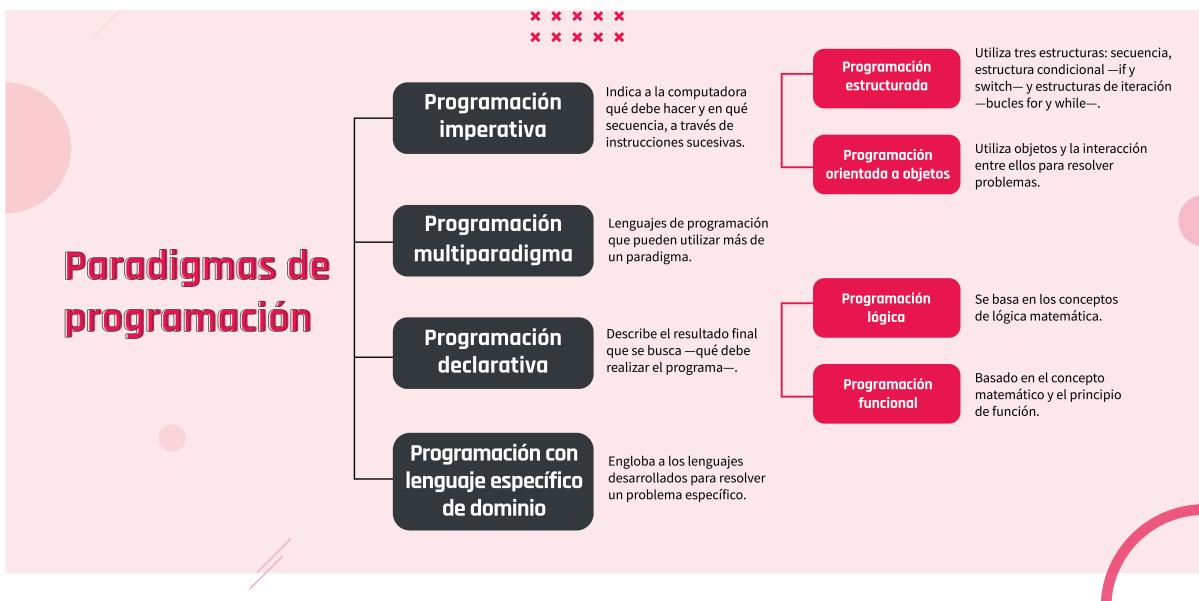
En JavaScript se puede escribir código tanto con el paradigma estructurado como con programación orientada a objetos e incluso utilizar el paradigma funcional.

## ¿Esto quiere decir que mientras más paradigmas tenga un lenguaje es mejor?

No, un lenguaje es una herramienta y hay distintas herramientas para distintas soluciones. Siempre debemos analizar el contexto, tiempos, con que equipo contamos, ¿hay presupuesto? ¿Cuáles son las herramientas que disponemos para trabajar? ¿Qué queremos lograr?



La mejor manera de conocer un paradigma de programación es investigar y programar en un lenguaje característico de ese paradigma. No hace falta ser un experto. Solo el hecho de conocerlo nos brinda más herramientas a la hora de desarrollar.



## Del código al ejecutable

“

El primer **compilador de la historia**, el **A-0**, fue desarrollado en 1952 por la científica en computación **Grace Hopper**.



”

Del código al ejecutable

DigitalHouse >  
Coding School

## Código Fuente

El código fuente es una colección de instrucciones de computadora escritas usando un lenguaje de programación legible por humanos.

```
40
41
42 $(function(){cards();});
43 $(window).on('resize', function(){cards();});
44 ▶ function cards(){
45   var width = $(window).width();
46   ▶ if(width < 750){
47     cardssmallscreen();
48   } else{
49     cardsbigscreen();
50   }
51 ▶ function cardssmallscreen(){
52   var cards = $('.card').length;
53   var height = 0;
54   var cardz = 2;
55   ▶ if(cards < cardz){height = 1;
56   cardz = 1;
57   }
58   var card = $(
59     '

' +
60       '

Card ' + cardz + '

' +
61     '

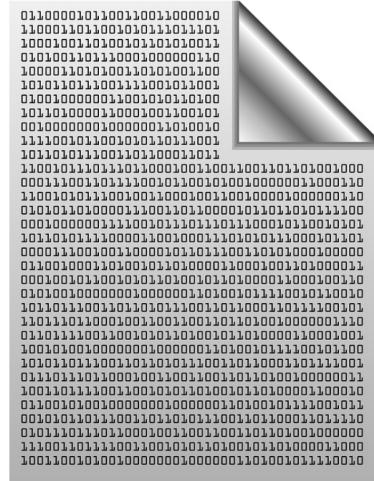
' +
62   );
63   card.appendTo('.grid');
64 }
```

Del código al ejecutable

DigitalHouse >  
Coding School

# Código de máquina

El código de máquina es una secuencia de sentencias en lenguaje de máquina o binario. Es el resultado obtenido después de que el compilador convierta el código fuente en un lenguaje que pueda ser comprendido por el procesador.

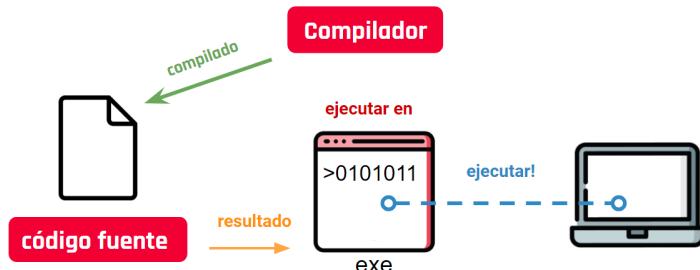


Del código al ejecutable

DigitalHouse >  
Coding School

# Compilador

Es una aplicación traduce (compila) el código fuente en un código que el procesador puede comprender y ejecutar. Este código de máquina se almacena en forma de archivo ejecutable.

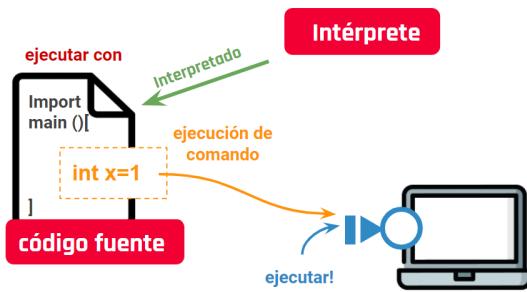


## Del código al ejecutable

DigitalHouse >  
Coding School

# Intérprete

Traduce el código fuente línea a línea y lo ejecuta directamente. El proceso de traducción funciona mucho más rápido que en un compilador, pero la ejecución es más lenta y se necesita una gran cantidad de memoria.



Del código al ejecutable

DigitalHouse >  
Coding School